X-ray Anomaly Detection in Industrial Pipelines

Diamantis Rafail Papadam, Christos Papaioannidis, Alexandros Zamioudis, and Ioannis Pitas
Department of Informatics, Aristotle University of Thessaloniki, Greece
pitas@csd.auth.gr

ABSTRACT

As Deep Neural Network (DNN)-based algorithms are improving, pivotal changes are happening towards efficient and effective automation in the field of industrial inspection. In the scope of our project, we analyze X-ray images of steel pipelines to detect the presence of corrosion in a novel way. In our industrial scenario, a drone lands a crawler that is equipped with an X-ray system on top of insulated pipelines to perform X-ray scans which are able to penetrate only the insulation, due to power consumption limitations. In this paper, we use modern unsupervised anomaly detection algorithms to detect the presence of corrosion, and the results are quite promising. Moreover, to compare several state-of-the-art approaches in terms of robustness to noise, we simulate two types of noise that can occur: (i) Poisson Noise, (ii) Motion Blur Noise. We conclude that the problem we are dealing with can be handled sufficiently well with state-of-the-art approaches, and that in the scenario of noise, the most robust algorithms are based on memory banks and teacher-student architectures.

Keywords: Deep Learning, Anomaly Detection, X-ray, Pipeline Defects, Industrial Automation

1. INTRODUCTION

This paper is part of a broader project, which aims at reducing human workload and stress in industrial pipeline inspection, while also increasing safety. In particular, an autonomous drone is tasked to land a crawler on top of industrial insulated pipelines. The crawler is equipped with an X-ray system which is able to penetrate the pipe insulation but not the steel that the pipes are made of, due to power consumption limitations. The X-ray system performs radiography scans, which are the focus of this paper. Specifically, we deal with the task of detecting corrosion by analysing such X-ray scans. Before moving forward, we list the main advantages of this type of inspection system: (i) Non destructive testing (NDT), since the insulation does not need to be removed, (ii) The crawler can access hard-to-reach areas, (iii) Human workers' safety and reduced workload.

Anomaly detection (AD) entails detecting and localizing unusual/unexpected patterns in the input data, without having any prior knowledge about these patterns. Such algorithms play a crucial role in many medical and industrial applications, such as tumor detection [33], defect detection in production lines [18], and damage detection in X-ray scans [9]. In the scope of this paper, being able to automatically detect damages/defects in insulated industrial pipes from X-ray scans is very important for ensuring the safe operation of oil and gas factories, while simultaneously reducing workload, and injury risk for human workers.

With the groundbreaking advancements of Deep Neural Network (DNN)-based algorithms over the past decade, automated AD has achieved remarkable results in real-world scenarios [15, 29, 34]. One approach to deal with AD using DNNs is by using supervised learning, where all types of damage have to be known and annotated in advance for the algorithm to be trained. As highlighted in [31], this is not only time-consuming but it also assumes that any type of defect that might occur in the future is known a priori, which is hardly ever the case in real-world applications. Moreover, in this approach, annotation noise that leads to degraded performance is almost inevitable.

In order to tackle the abovementioned issues, unsupervised AD was introduced, and today it constitutes a heavily researched topic [5, 23, 31]. In most of these approaches, a DNN is trained solely on normal data that contain no unexpected patterns. Then, during inference, when test data with any unexpected patterns is fed into the trained network, the latter is able to recognize it, and classify the data sample as anomalous.

Typically, the aforementioned methods have been employed for analyzing RGB images. However, in specific cases, RGB images are not available. For example, damage detection or corrosion detection in insulated pipelines is not possible by using RGB images, since damages/corrosion typically appear on the pipes below the insulation. In these cases, X-

ray data [22] is really useful, as it can effectively depict the pipe below the insulation, enabling damage/corrosion detection without the need of insulation removal by human workers.

To this end, we address the problem of insulated pipe corrosion detection using X-ray image data. Despite its broad interest to the industry, to the best of our knowledge, this is the first time that corrosion detection with X-ray data that do not penetrate the material of interest (*i.e.* steel pipes in our case) is considered as an anomaly detection problem. To this end, this paper offers the following main contributions: (i) Performance evaluation of state-of-the-art unsupervised anomaly detection algorithms in a novel, inadequately explored domain. (ii) Evaluation of such algorithms in terms of robustness to realistic noise.

2. RELATED WORK

This section is split in two parts. In the first part, we explore the AD literature which is used in many domains today. In the second part, we briefly build a background on the X-ray image processing domain.

2.1 Anomaly Detection

Anomaly detection is a heavily researched topic, with numerous different approaches. In a recent survey by Jiaqi Liu *et al.* [17], a taxonomy that categorizes unsupervised AD in two main categories is proposed: (i) Feature Embedding based Methods, (ii) Reconstruction based Methods. In the rest of this Section, we present the methods that we tested in the domain of pipeline X-ray corrosion detection, using this categorization.

2.1.1 Feature Embedding based Methods

Let us begin with a methodology based on a teacher-student architecture. In *STFPM* [32], a student-teacher framework is proposed where the teacher is a residual neural network (ResNet), pre-trained on ImageNet. The student has the same architecture as the teacher but its weights and biases are initialized randomly. During training, non-anomalous images are given as input and the student learns to match the feature maps of the teacher at different levels. During testing, the multi-level difference in the feature maps of the two networks is used to determine the anomaly score. Since the student is trained to match the teacher's behavior using normal images, the hypothesis is that when an abnormal image is fed into both networks the multi-level difference will increase significantly.

We continue our review with methodologies based on distribution maps. Interestingly, in *R-KDE* [1], the authors use an extra step before feature extraction which proposes regions with the use of Faster R-CNN [24]. Moreover, they extract features with the use of AlexNet [16]. Finally, they perform Kernel Density Estimation (KDE) [28] to model the normal features and detect anomalies during inference. Furthermore, in *DFM* [2], the distribution of the features extracted by a pre-trained classification DNN was modeled by: (i) Gaussian distribution, (ii) Gaussian mixture model. In this way, out-of-distribution detection is performed at inference time. To move to more modern approaches that incorporate Normalizing Flows (NFs) [20], in *C-Flow* [10], the authors work on a patch-level to achieve anomaly detection as well as localization. The patches are fed into a CNN feature extractor and the k feature maps that we have chosen to use for normal-sample distribution modeling are generated for each patch. Those feature maps, along with the positional encoding of the patches are passed as input in decoders which act as NFs and transform the feature distribution into a Gaussian distribution. In a slightly different manner, the authors of *FastFlow* [35], modeled the feature distribution by transforming it into a two-dimensional normal distribution, further increasing AD performance. Notably, in *U-Flow* [30], the authors propose a U-shape architecture which is based on the well-known U-Net paradigm [25]. In their work, the multi-scale feature extractor acts like the encoder, while the set of normalizing flows acts like the decoder. Furthermore, each normalizing flows is trained to map the normal data into a Gaussian Distribution. At inference time, anomaly maps are created for each scale, and they get upsampled to the original image size. Moreover, due to a guaranteed statistical independence described in the paper, the final pixel-level probability scores are calculated as the product of individual scores for each scale. Finally, estimation of the Number of False Alarms (NFA) [7], is used to produce the anomaly segmentation masks.

We conclude the review on feature embedding based methods with approaches that use a memory bank. In *DFKDE*, which is implemented in the *anomalib framework* [3], a CNN feature extractor is used to extract high level features. Moreover, PCA is performed to reduce the feature dimensionality, and the processed features are stored in a memory bank. As a final step, KDE [28] is used to model the normal data. Furthermore, in *PaDiM* [6], the input is indirectly split into patches, by considering the corresponding areas of a pre-trained ResNet's feature map, and a Gaussian distribution is created for each patch separately, by considering feature maps at different levels of the ResNet. This patch-level distribution creation with normal samples, allows for anomaly localization when an abnormal image is fed into the network. Finally, a method that focuses on industrial applications was introduced in *Patchcore* [26]. In a similar fashion to

PaDiM, image patches are considered, by passing the images into a pre-trained ResNet to extract features. Furthermore, the extracted features are sub-sampled and stored in a memory bank. Finally, during inference, the anomaly score for a test patch is calculated by performing a nearest neighbor search between the test sample features and the features stored in the memory bank.

2.1.2 Reconstruction based Methods

We only experimented with one reconstruction based method, namely *GANomaly* [4]. In this approach, the authors use a Generative Adversarial Network (GAN) to learn the distribution of normal samples within a latent space. Along with the training of the generator and the discriminator, an additional encoder that maps the reconstructed image to the same latent space is trained to match the vector in the generator's latent space. During inference, the distance between the encoder's output vector and the generator's latent space vector is used to determine the anomaly score.

2.2 X-ray Image Processing

In the first part of this section, we quickly review the industrial X-ray image processing literature, and in the second part, we briefly describe the two types of noise that we simulated in the scope of this paper.

2.2.1 Industrial X-ray Image Processing

Defect detection with the use of nondestructive testing (NDT) [8], has recently gained popularity due its effective inspection capabilities without causing any damage or altering the object of interest. Particularly, in the scope of this paper, we use radiography (X-ray) testing. A survey was conducted on this topic by Rafiei *et al.* [23], where X-ray image analysis, from traditional approaches to modern ones that make use of deep learning, is comprehensively presented.

In [21], Parlak and Emel, employ state-of-the-art object detection methods on an X-ray image dataset of aluminum parts which they also publish as openaccess. In their approach, the AD task is solved in a supervised learning manner. Recently, Intxausti *et al.* [13], show that self-supervised pre-training in X-ray data, allows for better feature extraction, which consistently yields better AD performance. However, due to the lack of sufficient publicly available X-ray data, most modern AD approaches simply use models pre-trained on ImageNet or other well-established classification datasets.

A similar domain to ours is that of Naddaf *et al.* [19], where a simple CNN architecture is developed for feature extraction, and a fully connected multilayer perceptron (MLP) that follows performs binary classification. However, the main difference is that the X-ray they use is able to penetrate the steel pipe, whereas our X-ray simply captures the edge between the pipe and the insulation while penetrating only the latter. Therefore, to the best of our knowledge there exists no other dataset in the literature like ours (Figure 1), while our novel X-ray system has several benefits which we have outlined in Section 1.

2.2.2 X-ray Image Noise

There are two main types of noise that can occur during an X-ray scan: Poisson Noise [27], and Motion Blur Noise [12]. In the scope of this paper, we test the robustness of state-of-the-art unsupervised AD algorithms to both types of noise separately, as well as a combination of the two.

3. METHODOLOGY

In this section, we present the approach we followed to evaluate different state-of-the-art algorithms in the novel domain of our project. In the first subsection, we present our raw dataset, and in the second subsection, we describe the process of noise simulation which is applied on the training set. Finally, in the third subsection, we illustrate the algorithm performance evaluation process we followed, using well-established metrics in the field of AD.

3.1 Dataset Description

Our dataset consists of 2300 normal images and 2000 abnormal images. All of the images are grayscale and each pixel is represented by 8 bits. Since we approach the problem as an unsupervised AD task, we use 300 normal images for the training set, and 2000 abnormal images along with the remaining 2000 normal images for the test set. The images are initially captured at several resolutions according to the quality setting. However, since high resolution is not important in our domain and most AD algorithms are very computationally-intensive, we resized all images to 112×112 pixels. In spite of this significant decrease in the resolution, as we will see in Section 4.1, several algorithms achieve an AUROC above 90% before simulated noise is added to the training set. Below, in Figure 1, we show an image from our project. On the lower part, we see the insulation which is penetrated by the X-ray, and on the upper part we see the steel pipe which has been affected by corrosion. If no corrosion were present, the edge between the steel pipe and the insulation

would closely resemble a straight line. To provide more context for our dataset, and given that the terms of our project prevent us from publishing it, we refer the reader to [22] which presents highly similar data.

Figure 1. X-ray image that depicts corrosion

3.2 Noise Simulation

To simulate both types of noise, we used Python v3.10.13. For the first type of noise, which is Poisson noise that typically occurs during X-ray scanning, we used the *numpy.random* library [11]. Below, we present the corresponding code snippet:

```
# Normalize the image to the range [0, 1]
img = numpy.asarray(img).astype(numpy.float32) / 255.0
# Apply Poisson noise to the image
noise_level = [1.00, 1.35, 1.70]
# Clip values to the range [0, 1] and scale back to [0, 255]
img = numpy.clip(img, 0, 1) * 255
```

For the second type of noise, which is horizontal motion blur that occurs particularly in our project since the X-ray crawler moves along the pipeline, we used the OpenCV library [14]. The corresponding code snippet is presented below:

```
# Create the motion blur kernel
kernel = np.zeros((size, size)) # size is one of: {15, 30, 45}
kernel[int((size-1) / 2), :] = np.ones(size) / size
# Apply the kernel
img = cv.filter2D(img, -1, kernel)
```

For both types of noise we simulated 3 levels of severity: low noise, medium noise, and high noise. Finally, we created a final training set which combined medium-level Poisson noise with medium-level motion blur noise. This means that we trained each algorithm in 8 training sets (1 training set without preprocessing, 3 training sets with Poisson noise, 3 training sets with motion blur noise, and 1 training set with combined medium-level noise).

3.3 Algorithm Performance Evaluation

The algorithm performance evaluation is done in two stages. Initially, we evaluate and compare the algorithms on our raw dataset that has not been preprocessed. Furthermore, to assess the robustness of each algorithm to noise that can realistically occur in our domain, we train each algorithm on the 7 preprocessed training sets, and test on the same test set as before. To perform our experiments, we made use of the anomalib framework [3]. In terms of hardware, we used an *Intel(R) Core(TM) i9-10900X* CPU, and a *12GB NVIDIA GeForce RTX 2080 Ti* GPU.

4. RESULTS

In this section we present our results in detail. Initially, we examine the performance of each algorithm in the dataset that has not been processed with simulated noise. Furthermore, we continue by examining the robustness of each algorithm to Poisson noise and to motion blur noise. Finally, we combine the two noise types to perform a final evaluation.

In each table of this section, we use **bold style** to denote the best value of each metric, and underline style to denote the second best. In the case of a draw, we use the corresponding style multiple times. Finally, as a main algorithm evaluation metric we consider the well-established *AUROC*. Since the normal and abnormal images in the test set are balanced, the second most useful metric is the F_1 *Score*. In Section 4.2 and Section 4.3, we plot the variability of these two metrics with respect to the amount of noise, to clearly illustrate which algorithms are more robust.

4.1 Algorithm Evaluation without Noise

In Table 1, we present the performance of 10 state-of-the-art algorithms in the scenario where there is no noise. The best algorithm according to the *AUROC* metric is U-Flow [30]. Moreover, we see how all algorithms except C-Flow,

GANomaly, and R-KDE, achieve an *AUROC* above 90%. The main finding of this section, is that several state-of-the-art algorithms are capable of effectively dealing with our task in the scenario where the X-ray images have not been affected by noise. Since our approach of collecting X-ray data is novel and no similar dataset is to the best of our knowledge used in the literature, we consider this to be the first main finding of this paper. In the following sections, we experiment with different types and levels of noise using the 7 algorithms that surpass 90% *AUROC*, to evaluate each of them in terms of robustness to noise, which constitutes the second and final main finding.

Table 1: Algorithm Performance without Noise

Algorithm	***AUROC***	***F_1 Score***	***Accuracy***	***Precision***	***Recall***
C-Flow	0.516	0.669	0.513	0.507	**0.988**
DFKDE	0.959	0.911	0.905	0.856	0.975
DFM	0.952	0.896	0.895	0.879	0.914
FastFlow	0.969	0.918	0.917	0.904	0.933
GANomaly	0.833	0.789	0.788	0.785	0.794
PaDiM	0.915	0.889	0.879	0.819	0.974
Patchcore	0.983	0.945	0.945	0.939	0.951
R-KDE	0.843	0.826	0.805	0.744	0.930
STFPM	0.962	0.909	0.905	0.875	0.945
U-Flow	**0.991**	**0.948**	**0.948**	**0.949**	0.947

4.2 Algorithm Evaluation under Poisson Noise

As shown in Table 2, we compare the algorithms under low, medium, and high Poisson noise. By looking at the *AUROC* metric, we see that the best 3 algorithms, regardless of the noise level (low/medium/high), are in descending order: Patchcore, STFPM, and DFKDE. Although in the previous section we identified that U-Flow is the best algorithm without simulated noise, it performs poorly when Poisson noise is introduced.

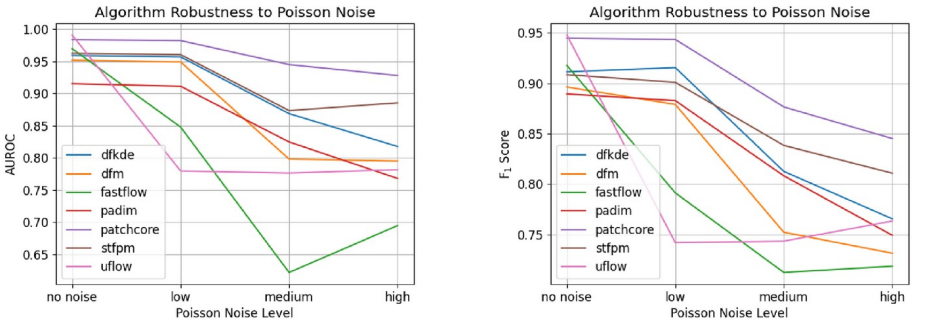
Moreover, in the same table that, as visualized in Figure 2, we measure how much the performance drops compared to the no-noise scenario, to be able to examine the robustness of each algorithm to Poisson noise. In particular, we divide the performance of each value in Table 2, with the corresponding performance shown in Table 1. Unexpectedly, in the low Poisson noise scenario, DFKDE's performance very slightly increases compared to the no-noise scenario. However, when Poisson noise is increased to medium and high level, performance drops as expected. We observe that the 3 most robust algorithms in descending order, are consistently: Patchcore, STFPM, DFKDE. Therefore, these algorithms not only perform best under Poisson noise, but they also demonstrate the highest robustness to it.

From the observations we made, we conclude that the most suitable approaches to deal with Poisson noise are either based on a memory bank (Patchcore, DFKDE), or on a teacher-student architecture (STFPM).

Table 2: Algorithm Performance & Robustness with Poisson Noise (low / medium / high)

Algorithm	***AUROC***	***F_1 Score***	***Accuracy***
DFKDE	Performance: (0.957 / 0.869 / 0.818) Robustness: (99.8% / 90.6% / 85.3%)	Performance: (0.916 / 0.813 / 0.765) Robustness: (**100.5**% / 89.2% / 84%)	Performance: (0.911 / 0.786 / 0.734) Robustness: (**100.7**% / 86.9% / 81.1%)
DFM	Performance: (0.949 / 0.798 / 0.795) Robustness: (99.7% / 83.8% / 83.5%)	Performance: (0.879 / 0.752 / 0.731) Robustness: (98.1% / 83.9% / 81.6%)	Performance: (0.881 / 0.709 / 0.698) Robustness: (98.4% / 79.2% / 78%)
FastFlow	Performance: (0.848 / 0.622 / 0.695) Robustness: (87.5% / 64.2% / 71.7%)	Performance: (0.791 / 0.712 / 0.718) Robustness: (86.2% / 77.6% / 78.2%)	Performance: (0.749 / 0.623 / 0.636) Robustness: (81.7% / 67.9% / 69.4%)
PaDiM	Performance: (0.911 / 0.825 / 0.768) Robustness: (99.6% / 90.2% / 83.9%)	Performance: (0.883 / 0.808 / 0.749) Robustness: (99.3% / 90.9% / 84.3%)	Performance: (0.877 / 0.784 / 0.724) Robustness: (99.8% / 89.2% / 82.4%)
Patchcore	Performance: (**0.982** / **0.945** / **0.928**) Robustness: (**99.9**% / **96.1**% / **94.4**%)	Performance: (**0.944** / **0.877** / **0.845**) Robustness: (99.9% / **92.8**% / **89.4**%)	Performance: (**0.943** / **0.869** / **0.833**) Robustness: (99.8% / **92**% / **88.1**%)
STFPM	Performance: (0.960 / 0.873 / 0.885) Robustness: (99.8% / 90.7% / 92%)	Performance: (0.901 / 0.838 / 0.811) Robustness: (99.1% / 92.2% / 89.2%)	Performance: (0.898 / 0.814 / 0.779) Robustness: (99.2% / 89.9% / 86.1%)

U-Flow	Performance: (0.780 / 0.776 / 0.782) Robustness: (78.7% / 78.3% / 78.9%)	Performance: (0.742 / 0.743 / 0.763) Robustness: (78.3% / 78.4% / 80.5%)	Performance: (0.694 / 0.690 / 0.721) Robustness: (73.2% / 72.8% / 76.1%)



(a) AUROC with respect to Poisson Noise **(b)** F_1 Score with respect to Poisson Noise

Figure 2. Robustness of Algorithms to Poisson Noise

4.3 Algorithm Evaluation under Motion Blur Noise

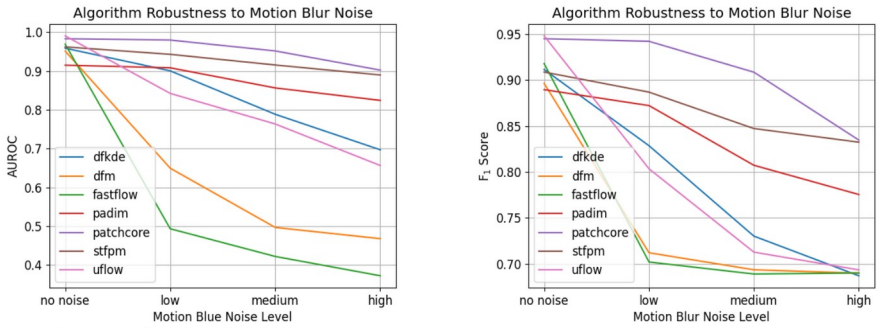
Similarly to the previous section, in Table 3, we compare the algorithm performance under motion blur noise. We notice that the best algorithms, regardless of the noise level (low/medium/high), are in descending order: Patchcore, STFPM, PaDiM.

Furthermore, in the same table, as is visualized in Figure 3, we compare the algorithms in terms of robustness to motion blur noise. In this comparison, we do not have a consistent order of the top 3 algorithms at different noise levels. However, in all noise levels, the best 3 algorithms are (in random order): Patchcore, STFPM, PaDiM. Hence, the algorithms that demonstrate the best performance under motion blur noise, also demonstrate the best robustness to it.

From the aforementioned observations, we conclude that in the scenario of motion blur noise the most suitable approaches are, similarly to Section 4.2, either based on a memory bank (Patchcore, PaDiM), or on a teacher-student architecture (STFPM).

Table 3: Algorithm Performance & Robustness with Motion Blur Noise (low / medium / high)

Algorithm	***AUROC***	***F_1 Score***	***Accuracy***
DFKDE	Performance: (0.901 / 0.789 / 0.697) Robustness: (94% / 82.3% / 72.7%)	Performance: (0.828 / 0.730 / 0.687) Robustness: (90.9% / 80.1% / 75.4%)	Performance: (0.814 / 0.703 / 0.653) Robustness: (89.9% / 77.7% / 72.2%)
DFM	Performance: (0.650 / 0.497 / 0.468) Robustness: (68.3% / 52.2% / 49.2%)	Performance: (0.712 / 0.694 / 0.690) Robustness: (79.5% / 77.5% / 77%)	Performance: (0.601 / 0.566 / 0.559) Robustness: (67.2% / 63.2% / 62.5%)
FastFlow	Performance: (0.494 / 0.422 / 0.372) Robustness: (51% / 43.6% / 38.4%)	Performance: (0.702 / 0.689 / 0.690) Robustness: (76.5% / 75.1% / 75.2%)	Performance: (0.585 / 0.558 / 0.556) Robustness: (63.8% / 60.9% / 60.6%)
PaDiM	Performance: (0.908 / 0.857 / 0.825) Robustness: (99.2% / 93.7% / 90.2%)	Performance: (0.872 / 0.807 / 0.775) Robustness: (98.1% / 90.8% / 87.2%)	Performance: (0.868 / 0.793 / 0.761) Robustness: (98.7% / 90.2% / 86.6%)
Patchcore	Performance: (**0.980** / **0.952** / **0.903**) Robustness: (**99.7**% / **96.8**% / 91.9%)	Performance: (**0.942** / **0.908** / **0.835**) Robustness: (**99.7**% / **96.1**% / 88.4%)	Performance: (**0.942** / **0.909** / **0.837**) Robustness: (**99.7**% / **96.2**% / 88.6%)
STFPM	Performance: (0.943 / 0.916 / 0.890) Robustness: (98% / 95.2% / **92.5**%)	Performance: (0.887 / 0.847 / 0.832) Robustness: (97.6% / 93.2% / **91.5**%)	Performance: (0.882 / 0.849 / 0.827) Robustness: (97.5% / 93.8% / **91.4**%)
U-Flow	Performance: (0.842 / 0.764 / 0.657) Robustness: (85% / 77.1% / 66.3%)	Performance: (0.803 / 0.713 / 0.694) Robustness: (84.7% / 75.2% / 73.2%)	Performance: (0.815 / 0.708 / 0.563) Robustness: (86% / 74.7% / 59.4%)



(a) AUROC with respect to Motion Blur Noise **(b)** F_1 Score with respect to Motion Blur Noise

Figure 3. Robustness of Algorithms to Poisson Noise

4.4 Algorithm Evaluation under Combined Noise

As a final algorithm evaluation, we run tests with combined noise (medium-level Poisson noise + medium-level motion blur noise). In Table 4, we see the algorithm performance and robustness under the combined noise. We observe that the best 3 algorithms in descending order are: Patchcore, DFKDE, PaDiM. Furthermore, the 3 most robust algorithms in descending order are: Patchcore, PaDiM, DFKDE. Overall, we empirically found that the most suitable algorithms for the combined noise scenario are based on a memory bank (Patchcore, PaDiM, DFKDE).

Table 4: Algorithm Performance & Robustness with medium-level Combined Noise

Algorithm	***AUROC***	***F_1 Score***	***Accuracy***
DFKDE	Performance: 0.804 Robustness: 83.8%	Performance: 0.756 Robustness: 83%	Performance: 0.719 Robustness: 79.4%
DFM	Performance: 0.542 Robustness: 56.9%	Performance: 0.689 Robustness: 76.9%	Performance: 0.549 Robustness: 61.3%
FastFlow	Performance: 0.428 Robustness: 44.2%	Performance: 0.682 Robustness: 74.3%	Performance: 0.543 Robustness: 59.2%
PaDiM	Performance: 0.795 Robustness: 86.9%	Performance: 0.795 Robustness: 89.4%	Performance: 0.771 Robustness: 87.7%
Patchcore	Performance: **0.900** Robustness: **91.6**%	Performance: **0.847** Robustness: **89.6**%	Performance: **0.843** Robustness: **89.2**%
STFPM	Performance: 0.745 Robustness: 77.4%	Performance: 0.751 Robustness: 82.6%	Performance: 0.691 Robustness: 76.4%
U-Flow	Performance: 0.695 Robustness: 70.1%	Performance: 0.686 Robustness: 72.4%	Performance: 0.546 Robustness: 57.6%

5. CONCLUSIONS

To conclude, in the scope of this paper, we implemented state-of-the-art unsupervised anomaly detection algorithms in a novel domain which, to the best of our knowledge, has not been publicly researched in the past. Initially, we found that most state-of-the-art unsupervised AD algorithms are capable of effectively solving our problem. Furthermore, we simulated 2 types of noise that can realistically occur in our domain, and found that the most suitable algorithms, both in terms of performance and robustness to such noise, are based on: (i) memory bank approaches, (ii) teacher-student architecture approaches.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon research and innovation programme under grant agreement number 101070604 (SIMAR).

REFERENCES

[1] Adey, P., Hamilton, O., Bordewich, M., Breckon, T.: Region based anomaly detection with real-time training and analysis. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). pp. 495–499 (2019). https://doi.org/10.1109/ICMLA.2019.00092

[2] Ahuja, N.A., Ndiour, I., Kalyanpur, T., Tickoo, O.: Probabilistic modeling of deep features for out-of-distribution and adversarial detection. arXiv preprint arXiv:1909.11786 (2019). https://doi.org/10.48550/arXiv.1909.11786

[3] Akcay, S., Ameln, D., Vaidya, A., Lakshmanan, B., Ahuja, N., Genc, U.: Anomalib: A deep learning library for anomaly detection. In: 2022 IEEE International Conference on Image Processing (ICIP). pp. 1706–1710 (2022). https://doi.org/10.1109/ICIP46576.2022.9897283

[4] Akcay, S., Atapour-Abarghouei, A., Breckon, T.P.: Ganomaly: Semi-supervised anomaly detection via adversarial training. In: Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14. pp. 622–637. Springer (2019). https://doi.org/10.1007/978-3-030-20893-6_39

[5] Cui, Y., Liu, Z., Lian, S.: A survey on unsupervised anomaly detection algorithms for industrial images. IEEE Access 11, 55297–55315 (2023). https://doi.org/10.1109/ACCESS.2023.3282993

[6] Defard, T., Setkov, A., Loesch, A., Audigier, R.: Padim: a patch distribution modeling framework for anomaly detection and localization. In: International Conference on Pattern Recognition. pp. 475–489. Springer (2021). https://doi.org/10.1007/978-3-030-68799-1_35

[7] Desolneux, A., Moisan, L., Morel, J.M.: From gestalt theory to image analysis: a probabilistic approach, vol. 34. Springer Science & Business Media (2007). https://doi.org/10.1007/978-0-387-74378-3

[8] Dwivedi, S.K., Vishwakarma, M., Soni, A.: Advances and researches on non destructive testing: A review. Materials Today: Proceedings 5(2), 3690–3698 (2018). https://doi.org/10.1016/j.matpr.2017.11.620

[9] Gong, Y., Luo, J., Shao, H., Li, Z.: A transfer learning object detection model for defects detection in x-ray images of spacecraft composite structures. Composite Structures 284, 115136 (2022). https://doi.org/10.1016/j.compstruct.2021.115136

[10] Gudovskiy, D., Ishizaka, S., Kozuka, K.: CFLOW-AD: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 98–107 (January 2022). https://doi.org/10.1109/WACV51458.2022.00188

[11] Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature 585(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2

[12] Huda, W., Abrahams, R.B.: Radiographic techniques, contrast, and noise in xray imaging. American Journal of Roentgenology 204(2), W126–W131 (2015). https://doi.org/10.2214/AJR.14.13116

[13] Intxausti, E., Skočaj, D., Cernuda, C., Zugasti, E.: A methodology for advanced manufacturing defect detection through self-supervised learning on x-ray images. Applied Sciences 14(7), 2785 (2024). https://doi.org/10.3390/app14072785

[14] Itseez: Open source computer vision library. https://github.com/itseez/opencv (2015)

[15] Jiang, Y., Wang, W., Zhao, C.: A machine vision-based realtime anomaly detection method for industrial products using deep learning. In: 2019 Chinese Automation Congress (CAC). pp. 4842–4847 (2019). https://doi.org/10.1109/CAC48633.2019.8997079

[16] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25 (2012), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[17] Liu, J., Xie, G., Wang, J., Li, S., Wang, C., Zheng, F., Jin, Y.: Deep industrial image anomaly detection: A survey. Machine Intelligence Research 21(1), 104–135 (2024). https://doi.org/10.1007/s11633-023-1459-z

[18] Lu, B., Xu, D., Huang, B.: Deep-learning-based anomaly detection for lace defect inspection employing videos in production line. Advanced Engineering Informatics 51, 101471 (2022). https://doi.org/10.1016/j.aei.2021.101471

[19] Naddaf-Sh, M.M., Naddaf-Sh, S., Zargaradeh, H., Zahiri, S.M., Dalton, M., Elpers, G., Kashani, A.R.: Next-generation of weld quality assessment using deep learning and digital radiography. In: Proceedings of the AAAI Spring Symposium Series (2020), https://www.researchgate.net/publication/340234164_Next-Generation_of_Weld_Quality_Assessment_Using_Deep_Learning_and_Digital_Radiography

[20] Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. Journal of Machine Learning Research 22(57), 1–64 (2021), http://jmlr.org/papers/v22/19-1028.html

[21] Parlak, I.E., Emel, E.: Deep learning-based detection of aluminum casting defects and their types. Engineering Applications of Artificial Intelligence 118, 105636 (2023). https://doi.org/10.1016/j.engappai.2022.105636

[22] QSA GLOBAL: OpenVision HD Demo. https://www.qsa-global.com/openvision-hd

[23] Rafiei, M., Raitoharju, J., Iosifidis, A.: Computer vision on x-ray data in industrial production and security applications: A comprehensive survey. Ieee Access 11, 2445–2477 (2023). https://doi.org/10.1109/ACCESS.2023.3234187

[24] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems 28 (2015), https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[25] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

[26] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., Gehler, P.: Towards total recall in industrial anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14318–14328 (2022). https://doi.org/10.1109/CVPR52688.2022.01392

[27] Sai, G.V., Seshank, C., Krishna, P.P.S., Dhatterwal, J.S.: Reduction of noise in medical imaging quality. In: 2023 International Conference on Disruptive Technologies (ICDT). pp. 364–368. IEEE (2023). https://doi.org/10.1109/ICDT57929.2023.10150846

[28] Scott, D.W.: Multivariate Density Estimation and Visualization, pp. 549–569. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-21551-3_19

[29] Siegel, B.: Industrial anomaly detection: A comparison of unsupervised neural network architectures. IEEE Sensors Letters 4(8), 1–4 (2020). https://doi.org/10.1109/LSENS.2020.3007880

[30] Tailanian, M., Pardo, Á., Musé, P.: U-flow: A u-shaped normalizing flow for anomaly detection with unsupervised threshold. Journal of Mathematical Imaging and Vision pp. 1–19 (2024). https://doi.org/10.1007/s10851-024-01193-y

[31] Tao, X., Gong, X., Zhang, X., Yan, S., Adak, C.: Deep learning for unsupervised anomaly localization in industrial images: A survey. IEEE Transactions on Instrumentation and Measurement (2022). https://doi.org/10.1109/TIM.2022.3196436

[32] Wang, G., Han, S., Ding, E., Huang, D.: Student-teacher feature pyramid matching for anomaly detection. arXiv preprint arXiv:2103.04257 (2021). https://doi.org/10.48550/arXiv.2103.04257

[33] Wang, N., Chen, C., Xie, Y., Ma, L.: Brain tumor anomaly detection via latent regularized adversarial network. arXiv preprint arXiv:2007.04734 (2020). https://doi.org/10.48550/arXiv.2007.04734

[34] Wang, W., Wang, Z., Zhou, Z., Deng, H., Zhao, W., Wang, C., Guo, Y.: Anomaly detection of industrial control systems based on transfer learning. Tsinghua Science and Technology 26(6), 821–832 (2021). https://doi.org/10.26599/TST.2020.9010041

[35] Yu, J., Zheng, Y., Wang, X., Li, W., Wu, Y., Zhao, R., Wu, L.: Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. arXiv preprint arXiv:2111.07677 (2021). https://doi.org/10.48550/arXiv.2111.07677